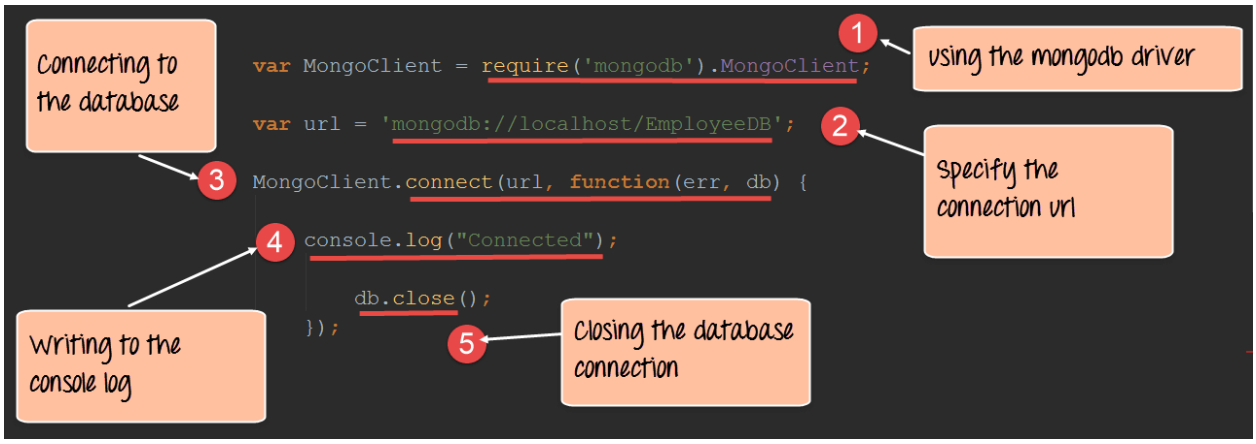
Date⇒ 29-01-2021
Module⇒ Backend
Lecture By⇒ Akash Handa
Subject ⇒ Connecting Nodejs With mongodb

IN PREVIOUS LECTURE **(QUICK RECAP) Date-28/1/2021**	**In Today's Lecture (Overview)**
What Is NoSql Database What is Mongodb How To Setup Mongodb Mongodb Queries/Commands	How to Connect Node js File with mongo data base Import MongoClient Create our main function

How to Connect Node js File with mongo data base

⇒ First of all start Your mongo Database



Import MongoClient

The MongoDB module exports MongoClient, and that's what we'll use to connect to a MongoDB database. We can use an instance of MongoClient to connect to a cluster, access the database in that cluster, and close the connection to that cluster.
const {MongoClient} = require('mongodb');

Create our main function

Let's create an asynchronous function named main() where we will connect to our MongoDB cluster, call functions that query our database, and disconnect from our cluster.
The first thing we need to do inside of main() is create a constant for our connection URI. The connection URI is the connection string you copied in Atlas in the previous section. When you paste the connection string, don't forget to update <username> and <password> to be the credentials for the user you created in the previous section. Note: the username and password you provide in the connection string are NOT the same as your Atlas credentials.

```
/**
* Connection URI. Update <username>, <password>, and <your-cluster-url> to
reflect your cluster.
* See https://docs.mongodb.com/ecosystem/drivers/node/ for more details
*/
const uri =
"mongodb+srv://<username>:<password>@<your-cluster-url>/test?retryWrites=t
rue&w=majority";
```

Now that we have our URI, we can create an instance of MongoClient.

```
const client = new MongoClient(uri);
```

Note: When you run this code, you may see DeprecationWarnings around the URL string parser and the Server Discover and Monitoring engine. If you see these warnings, you can remove them by passing options to the MongoClient. For example, you could instantiate MongoClient by calling new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true }). See the Node.js MongoDB Driver API documentation for more information on these options.
Now we're ready to use MongoClient to connect to our cluster. client.connect() will return a promise. We will use the await keyword when we call client.connect() to indicate that we should block further execution until that operation has completed.

```
await client.connect();
```

Now we are ready to interact with our database. Let's build a function that prints the names of the databases in this cluster. It's often useful to contain this logic in well named functions in order to improve the readability of your codebase. Throughout this series, we'll create new functions similar to the function we're creating here as we learn how to write different types of queries. For now, let's call a function named listDatabases().

```
await listDatabases(client);
```

Once we have our main() function written, we need to call it. Let's send the errors to the console.

```
main().catch(console.error);
```

Putting it all together, our main() function and our call to it will look something like the following.

```
async function main(){
    /**
     * Connection URI. Update <username>, <password>, and
<your-cluster-url> to reflect your cluster.
     * See https://docs.mongodb.com/ecosystem/drivers/node/ for more
details
     */
    const uri =
"mongodb+srv://<username>:<password>@<your-cluster-url>/test?retryWrites=t
rue&w=majority";


    const client = new MongoClient(uri);

    try {
        // Connect to the MongoDB cluster
        await client.connect();

        // Make the appropriate DB calls
        await  listDatabases(client);

    } catch (e) {
        console.error(e);
    } finally {
        await client.close();
    }
}

main().catch(console.error);
```

List the databases in our cluster

In the previous section, we referenced the listDatabases() function. Let's implement it!

This function will retrieve a list of databases in our cluster and print the results in the console.

```
async function listDatabases(client){
    databasesList = await client.db().admin().listDatabases();

    console.log("Databases:");
    databasesList.databases.forEach(db => console.log(` - ${db.name}`));
};
```